비트 클러스터링을 이용한 빈발 패턴 탐사의 성능 개선 방안

Advanced Improvement for Frequent Pattern Mining using Bit-Clustering

김의찬* / Euichan Kim **김계현**** / Kyehyun Kim
이철용*** / Chulyong Lee **박은지****** / Eunji Park

요 약

데이터마이닝은 데이터베이스에 저장되어 있는 많은 일반적인 정보들을 가지고 의미있는 정보를 찾아내는 것이다. 많은 데이터 마이닝 기법들 중에 클러스터링과 연관규칙을 다루는 연구가 많이 이뤄지고 있다. 클러스터링 기법에는 공간데이터를 다루거나 속성데이터(비공간 데이터)를 다루는 많은 기법들이 연구되고 있고, 연관규칙 또한 빈발 패턴을 찾아내는 연구가 활발히 진행되고 있다. 기존의 연구 중 apriori 연관규칙 알고리즘을 개선하는 방법으로 비트 클러스터링을 이용하는 방법이 있다. 우리는 apriori 연관규칙 보다 더 나은 성능을 나타내는 FP-Growth에 대해 살펴보고 FP-Growth의 문제점을 찾아 이를 해결하기 위한 방법으로 비트 클러스터링을 이용하여 해결할 수 있는지에 대해 연구하였다. 본 논문에서는 전체 데이터베이스를 비트 클러스터링을 통해 몇 개의 클러스터로 나누어 FP-Growth 방법에 사용할 것을 제안하였다. 이렇게 하면 기존의 FP-Growth 방법보다 더 나은 성능을 가질 수 있으며 이를 증명하기 위한 실험을 수행하였다. 실험은 패턴 마이닝 연구에서 사용하는 chess 데이터를 이용하였으며, 최소지지도를 다르게 적용하면서 FP-Tree를 생성하는 실험을 하였다. 최소지지도가 높은 경우에는 기존의 방법과 비슷한 결과를 얻었지만 그 외 경우에는 기존의 방법보다 본 논문에서 제안하는 방법이 더 우수한 결과를 얻을 수 있었다. 본 논문의 주요 결론으로서 비트 클러스터링을 이용한 방법이 상대적으로 우수한 데이터 마이닝 방법임을 정리하였으며, 아울러 GML 데이터를 위한 비트 클러스터링의 적용방법론에 대하여도 논의하였다.

Abstract

Data mining extracts interesting knowledge from a large database. Among numerous data mining techniques, research work is primarily concentrated on clustering and association rules. The clustering technique of the active research topics mainly deals with analyzing spatial and attribute data. And, the technique of association rules deals with identifying frequent patterns. There was an advanced apriori algorithm using an existing bit-clustering

■ 논문접수 : 2007.4.30 ■ 심사완료 : 2007.6.25

* 교신저자 인하대학교 공과대학 지리정보공학과 GIS 연구실 연구원(spesmico@inha.ac.kr)
** 인하대학교 공과대학 지리정보공학과 GIS 연구실 교수(kyehyun@inha.ac.kr)
*** 인하대학교 공과대학 지리정보공학과 GIS 연구실 박사과정(khsakura82@inhaian.net)
**** 인하대학교 공과대학 지리정보공학과 GIS 연구실 석사과정(ethel_126@naver.com)

algorithm. In an effort to identify an alternative algorithm to improve apriori, we investigated FP-Growth and discussed the possibility of adopting bit-clustering as the alternative method to solve the problems with FP-Growth. FP-Growth using bit-clustering demonstrated better performance than the existing method. We used chess data in our experiments. Chess data were used in the pattern mining evaluation. We made a creation of FP-Tree with different minimum support values. In the case of high minimum support values, similar results that the existing techniques demonstrated were obtained. In other cases, however, the performance of the technique proposed in this paper showed better results in comparison with the existing technique. As a result, the technique proposed in this paper was considered to lead to higher performance. In addition, the method to apply bit-clustering to GML data was proposed.

주요어 : 데이터마이닝, 클러스터링 연관규칙, FP-Growth
Keyword : Data Mining, Clustering, Association Rules, FP-Growth

1. 서 론

우리가 사용하는 데이터베이스 내에는 많은 양의 데이터들이 포함되어 있으며 그러한 데이터들은 실세계의 데이터와 일치한다. 데이터베이스로부터 정보를 얻기 위하여 우리는 데이터베이스에 질의를 하게 되고 질의를 통해 얻어내는 정보 또한 실세계의 데이터와 일치하는 것들이다. 데이터베이스로부터 질의 검색을 통해 얻는 정보들은 기본적이고 일반적인 정보들인데, 데이터베이스로부터 기본적이고 일반적인 정보들이 아닌 함축적이고 암시적인 정보들을 얻고 싶은 경우가 있을 것이다. 이러한 함축적이고 암시적인 정보들을 얻는 방법이 데이터 마이닝(Data Mining)이다. 공간 데이터 마이닝은 데이터 마이닝을 공간으로 확장한 개념으로 공간 데이터베이스 상에 저장되어 있는 공간 데이터와 비공간 데이터(속성 데이터)를 이용하여 함축적이고 암시적인 정보들을 얻어내는 방법이다. 마이닝 기법에는 여러 가지 기법들이 있다. 연관규칙(Association Rules), 분류(Classification), 일반화(Generalization), 클러스터링(Clustering) 등 다양한 기법들이 있다[1, 2]. 이런 여러 기법들은 일반적인 분야뿐만 아니라 공간관련 분야까지 여러 분야에서 다양하게 연구되고 있다.

클러스터링 기법으로는 수치형이나 범주형으로 나눠볼 수 있는데 가장 잘 알려진 k-means 알고리즘, k-medoid 알고리즘 등이 있으며 그 외에도 다양한 기법들도 많이 나오고 있다[3, 4]. 각 알고리즘들은 공간 데이터와 비공간 데이터에 대한 마이닝에도 많이 사용되고 있다. 연관규칙 방법들 중에 잘 알려진 방법으로는 Apriori 기법[5]과 FP-Growth 기법[6]이 있다. 두 기법의 차이점은 후보 항목의 생성 여부이다. Apriori 기법은 후보 항목을 생성하여 빈발 항목을 찾아내지만 FP-Growth 기법은 후보 항목을 생성하지 않고 빈발트리를 생성하게 된다. 성능 부분에 있어서는 후보 항목을 생성하지 않고 처리가 되는 FP-Growth 기법이 우수한 것으로 나타났다.

본 논문에서는 FP-Growth의 성능을 개선하기 위하여 범주형 데이터를 다루는 비트 클러스터링[7] 응용에 대한 연구를 하였다. 기존의 연관규칙 방법 중 apriori 알고리즘을 개선하는 방법으로 비트 클러스터링을 이용하는 방법[7]이 있었다. 우리는 apriori보다 더 나은 성능을 보이는 FP-Growth의 문제점을 찾고 비트 클러스터링을 통해 개선할 수 있는 방법을 연구하였다. 전체 데이터베이스를 비트 클러스터링을 통해 몇 개의 클러스터로 나누어 FP-Growth를 적용하였을 때 기존의 FP-Growth 방법보다 더 나은 성능을 보인다는 것을 실험을 통해 확인하였다. 실험은 패턴 마이닝 연구에서 사용

하는 chess 데이터를 이용하였으며 최소지지도를 다르게 적용하였다. FP-Growth의 성능 대부분은 FP-Tree 생성이 좌우하기 때문에 FP-Tree를 생성하는 실험까지 하였다. 최소지지도가 높은 경우에는 기존의 방법과 비슷한 결과를 얻었지만 낮은 경우에는 기존의 방법보다 본 논문에서 제안하는 방법이 더 나은 성능을 갖는 결과를 얻었다. 이는 기존의 방법보다 본 논문에서 제안하는 방법이 더 우수하다는 것을 나타내준다. 추가적으로 GML 데이터에 비트 클러스터링을 어떻게 적용하여 마이닝할 수 있는지도 확인하였다.

본 논문의 구성은 2장에서는 본 연구의 기반이 되는 비트 클러스터링과 FP-Growth에 대하여 서술하였으며, 3장에서는 데이터에 대한 비트 클러스터링의 사용 예에 대하여 논의하였다. 4장에서는 본 연구의 실험 환경과 실험 결과를 통해 얻어낼 수 있는 정보에 대하여 기술하고 분석하였으며, 5장에서는 최종적인 결론을 정리하였다.

2. 관련 연구

2.1 비트 클러스터링(Bit Clustering)

클러스터링을 하는 데이터에는 크게 수치형 값과 범주형 값으로 나누어 볼 수 있다. 범주형 값의 경우는 클러스터를 형성하는 척도로 거리라는 개념을 적용하기가 쉽지 않은 경우다. 빈발 항목 개념을 이용한 트랜잭션 클러스터링 알고리즘은 범주형 값 중에서도 장바구니 데이터를 Apriori 알고리즘에서 사용한 빈발 항목 개념을 도입하여 빈발하는 아이템끼리만 클러스터링 한다[8, 9].

범주형 값들을 비교하는 방법에는 [10]에서도 언급되었던 범주형 값을 수치형 값으로 바꾼 다음 좌표값으로 인식하여 두 값 사이의 거리를 구하는 방법이 있다. 좌표값으로 인식하기 위해서 클러스터링을 하기 전에 각 트랜잭션의 아이템 종류를 보고 해당하는 아이템이 있으면 1, 없으면 0으로 표현한다. 이렇게 변환된 좌표를 유클리디안 거리법을 이용하여 인접한 것끼리 클러스터링을 하는 것이다.

[7]에서 제안한 비트 클러스터링에 대하여 살펴보면 다음과 같다. 비트 클러스터링을 이용하기 위해 먼저 데이터베이스에 있는 모든 트랜잭션들의 아이템 집합을 비트로 표현해야 한다. 비트로 표현된 데이터베이스를 이용하여 클러스터링을 시작한다. 비트 트랜잭션 클러스터링을 할 때는 2가지 식을 이용하게 된다. 첫 번째 식은 유사도 식으로 아래에 나와 있는 것과 같다.

$$t\_sim(t_i, t_j) = 1 - \frac{|xOR(t_i, t_j)|}{total\ num\ of\ items}$$

여기서, t_i, t_j는 트랜잭션을 의미하며 |xOR(t_i, t_j)|는 t_i와 t_j 트랜잭션을 비교하는데 해당 아이템의 유, 무 즉 0과 1을 비교하여 같지 않은 것의 개수를 나타내는 부분이다. 그것을 전체 아이템의 개수로 나누어 1에서 뺀 값이 두 트랜잭션의 유사도 값이 된다.

위의 유사도 식을 통하여 각 트랜잭션들 간의 유사도 값을 구하고, 유사도 값이 유사도 임계값보다 큰 것들끼리 묶어서 클러스터를 생성하게 된다. 그런데 유사도 값을 비교하는 과정에서 해당 클러스터의 모든 트랜잭션과 하나의 트랜잭션을 비교하는 것은 클러스터링 수행 성능을 떨어지게 한다. 따라서 해당 클러스터의 대푯값을 구하여 사용하는 것이 훨씬 더 효율적이다. 본 연구에서는 이러한 대푯값을 구하기 위하여 두 번째 식인 소유 가능도를 계산하게 된다. 소유 가능도는 아이템들이 임의의 트랜잭션에 포함될 가능성으로 정의할 수 있다. 소유 가능도 식은 다음과 같다.

$$pos(i_i) = \frac{|i_i|}{[i_i]}$$

여기서 i_i는 해당 아이템이 되며, $|i_i|$는 해당 아이템의 값이 1인 개수, $[i_i]$는 해당 아이템의 1과 0의 총 개수가 된다. 소유 가능도 식을 통해 소유 가능도 값을 구하고 난 다음 아래 정의들을 통해 해

당 아이템의 대푯값을 정하게 된다.

클러스터링의 대푯값을 구하기 위하여 다음과 같은 4개의 정의를 이용한다. 단, 소유 가능 임계값은 0.5를 초과하는 것으로 한다.(소유 가능 임계값 > 0.5)

[정의 1] *$pos(i_j)$ 값이 소유 가능 임계값보다 크거나 같으면 강한 소유 가능 아이템으로 간주한다.*

$pos(i_j)$ >= 소유 가능 임계값 → 강한 소유 가능 아이템(strengthen possessive item)

[정의 2] *$pos(i_j)$ 값이 1-소유 가능 임계값보다 작거나 같으면 약한 소유 가능 아이템으로 간주한다.*

$pos(i_j)$ <= 1-소유 가능 임계값 → 약한 소유 가능 아이템(weaken possessive item)

[정의 3] *$pos(i_j)$ 값이 1-소유 가능 임계값보다 크고 소유 가능 임계값보다 작으면 소유 가능 아이템으로 간주한다.*

1- 소유가능 임계값 < $pos(i_j)$ < 소유 가능 임계값 → 소유 가능 아이템(possessive item)

[정의 4] *해당 아이템이 강한 소유 가능 아이템이거나 소유 가능 아이템으로 간주되는 경우 해당 아이템의 대푯값을 1로 하고, 약한 소유 가능 아이템으로 간주되는 경우 해당 아이템의 대푯값을 0으로 한다.*

위의 4가지 정의를 이용하여 각 클러스터의 대푯값을 구하게 된다. 클러스터링의 전체적인 흐름은 다음과 같다.

1) 유사도 임계값과 소유 가능도 임계값을 입력받는다.

2) 각 트랜잭션들 간 또는 트랜잭션과 각 클러스터의 대푯값과의 유사도를 구한다.

3) 유사도 임계값보다 크거나 같으면 같은 클러스터로 포함시키고, 그렇지 않으면 다른 클러스터로 분류한다.

4) 클러스터에 새로운 트랜잭션이 포함되었다면 소유 가능도 식을 이용하여 클러스터의 대푯값을 구한다.

5) 2)에서 4)를 반복한다.

이러한 클러스터링에 대한 설명은 [11, 12, 13]에서도 잘 설명되고 있다.

2.2 FP-Growth(Frequent Pattern Growth)

Apriori 알고리즘 기법은 후보 항목 집합, 빈발 항목 집합을 생성하여 연관규칙을 찾아낸다. 그러나 상당한 크기의 후보 집합 생성이 필요할 수 있다. 이러한 후보 집합 생성을 하지 않고 빈발 항목 집합을 발견하기 위해 나온 방법이 분할-정보(divide-and-conquer) 기법을 사용하는 빈발 패턴 증가(Frequent Pattern Growth) 또는 간단히 FP-Growth라 하는 기법이다[6]. 이 방법은 빈발 항목을 가지는 데이터베이스를 빈발 패턴 트리(Frequent Pattern Tree: FP-Tree)로 압축하여 사용한다.

FP-Tree 생성 시에 전체 데이터베이스를 2번 검색하게 된다. 첫 번째 검색을 통해 빈발 아이템 리스트를 찾아낸다. 빈발 아이템 리스트는 주어진 최소 지지도 값보다 높은 아이템들을 모아둔 리스트이다. 이 리스트에 있는 빈발 아이템들의 지지도도 같이 저장하게 되는데 저장할 때는 지지도의 순서가 내림차순이 되도록 정렬을 시켜 저장한다. 빈발 아이템 리스트의 순서에 따라 두 번째 검색을 통하여 트랜잭션을 검색하면서 재귀적 호출을 통해 FP-Tree를 생성하게 된다. 그러나 FP-Tree를 생성할 때 걸리는 부하량이나 시간으로 인하여 성능저하를 일으키게 된다. 따라서 본 논문에서는 비트 클러스터링을 이용할 것을 제안한다.

3. 비트 클러스터링 사용 예

3.1 트랜잭션 DB와 FP-Growth에 비트 클러스터링 사용

· 비트 클러스터링을 하기 위해서는 먼저 해당 아이템들을 비트로 변환시켜줘야 한다. 즉, 아이템이 있으면 1, 없으면 0으로 변환시켜야 한다. 다음 <표 1>의 첫 번째에 있는 예제 데이터베이스를 비트로 표현하였을 때 두 번째 테이블과 같이 비트 표현 결과가 나오게 된다.

<표 1> 예제 데이터베이스

TID	ItemSet
T1	a, b, c, f
T2	a, b, d, e
T3	a, b, c, d, f, g
T4	a, b, d, f
T5	g, h, i, j
T6	d, e, g, h, j

TID	a	b	c	d	e	f	g	h	i	j
T1	1	1	1	0	0	1	0	0	0	0
T2	1	1	0	1	1	0	0	0	0	0
T3	1	1	1	1	0	1	1	0	0	0
T4	1	1	0	1	0	1	0	0	0	0
T5	0	0	0	0	0	0	1	1	1	1
T6	0	0	0	1	1	0	1	1	0	1

비트 클러스터링 과정은 <표 1>에 나와 있는 예제를 이용하여 설명하면 다음과 같다. 전체 트랜잭션의 개수는 6개이고 총 아이템의 개수는 10개이다. 2.1 절에서 언급한 대푯값 구하는 방법과 클러스터링 처리 흐름에 따라 클러스터링을 수행한다. 클러스터링을 할 때 사용한 유사도 임계값과 소유 가능도 임계값은 각각 0.6으로 가정하고 수행한 경우에 2개의 클러스터를 생성하게 된다.

첫 번째 트랜잭션은 첫 번째 클러스터에 포함되게 된다. 그 이유는 아직 생성된 클러스터가 없기 때문이다. 그리고 이렇게 하였을 때 첫 번째 클러스터에는 하나의 트랜잭션만 있게 되므로 대푯값을 구할 필요가 없다. 따라서 유사 가능도 계산을 하지 않고 첫 번째 트랜잭션 자체가 첫 번째 클러스터의 대푯값이 된다. 다음으로 두 번째 트랜잭션이 들어오고 첫 번째 클러스터의 대푯값과 유사도 계산을 하게 된다. 2.1절에서 언급한 유사도 식에 따라 유사도 계산을 하는데, 첫 번째 트랜잭션과 두 번째 트랜잭션의 $|xOR(t_1, t_2)|$ 값이 4가 나오게 되므로 $t_sim(t_1, t_2) = 1 - (|xOR(t_1, t_2)|/10) = 1 - (4/10) = 0.6$이 나온다. 유사도 임계값을 0.6으로 가정하였으므로 두 번째 트랜잭션은 첫 번째 클러스터에 포함되게 된다. 첫 번째 클러스터에 새로운 트랜잭션이 포함 되었으므로 첫 번째 클러스터의 대푯값을 구하여야 한다. 그림 1의 대푯값 구하는 알고리즘을 이용하여 각 아이템별로 소유 가능도를 구한다. 소유 가능도 식을 이용하면 아이템 a의 경우 $|i_a|$는 소유하고 있는 즉, 1의 개수가 되므로 2라는 값이 되며 $[i_a]$는 1과 0의 개수라고 했으므로 2가 된다. 계산하여 보면 2/2 = 1이라는 소유 가능도 값이 나오게 된다. 이는 소유 가능 임계값 0.6보다 크므로 [정의 1]에 따라 아이템 a는 소유 가능 아이템에 해당되며 [정의 4]를 통해 소유 가능 아이템의 경우 대푯값을 1로 놓는다고 하였으므로 아이템 a의 대푯값은 1이 된다. 아이템 c의 경우 $|i_c|$는 1, $[i_c]$는 2가 된다. 소유 가능도를 계산하여 보면 1/2 = 0.5가 되는데 소유 가능 임계값 0.6보다는 작지만 1-소유 가능 임계값보다는 큰 값이 된다. 그러므로 [정의 3]에 따라 아이템 c가 강한 소유 가능 아이템에 해당되게 된다. [정의 4]에서 강한 소유 가능 아이템의 경우 대푯값을 1로 놓는다고 하였으므로 아이템 c의 대푯값은 1이 되는 것이다. 다른 아이템들의 대푯값도 이와 같은 방법으로 구하게 되면 <표 2>와 같은 결과를 얻을 수 있다. 이는 첫 번째 클러스터의 대푯값이 된다.

<표 2> 첫 번째 클러스터의 대푯값

TID	a	b	c	d	e	f	g	h	i	j
T1	1	1	1	0	0	1	0	0	0	0
T2	1	1	0	1	1	0	0	0	0	0
C1	1	1	1	1	1	1	0	0	0	0

이러한 과정을 거치게 되면 그림 의 예제 데이터베이스는 <표 3>과 같이 2개의 클러스터를 얻게 된다.({T1, T2, T3, T4}, {T5, T6})

<표 3> 2개의 클러스터링 모습

TID	a	b	c	d	e	f	g	h	i	j
T1	1	1	1	0	0	1	0	0	0	0
T2	1	1	0	1	1	0	0	0	0	0
T3	1	1	1	1	0	1	1	0	0	0
T4	1	1	0	1	0	1	0	0	0	0

TID	a	b	c	d	e	f	g	h	i	j
T5	0	0	0	0	0	0	1	1	1	1
T6	0	0	0	1	1	0	1	1	0	1

전체 DB로부터 생성된 Tree와 각 클러스터들로부터 생성된Tree의 크기는 다르다. 최소지지도를 2로 가정하였을 때, 전체 DB로부터 얻어지는 아이템 리스트는 a, b, c, d, e, f, g, h, j이다. 그러나 각 클러스터로부터 얻어지는 아이템 리스트는 T1의 경우a, b, c, d, f이며, T2의 경우 g, h, j이다. 각 아이템 리스트를 통해서 FP-Tree를 생성하고 FP-Tree를 이용하여 규칙을 찾게 되는데, 비트 클러스터링을 하여 각각의 클러스터로 나눈 경우 아이템 e와 g에 대한 규칙이 제외된 것을 볼 수 있다. 약간의 손실을 가져오긴 하지만 <표 4>와 <표 5> 에서 보여주듯 거의 대부분의 규칙을 동일하게 얻어낼 수 있음을 알 수 있다.

<표 4> 전체 DB와 FP-Tree로부터 얻을 수 있는 규칙들

item	빈발 패턴 생성
j	(g j), (h j), (g h j)
h	(g h)
e	(d e)
c	(a c), (b c), (f c), (a b c), (a f c), (b f c), (a b f c)
g	(d g)
f	(a f), (b f), (d f), (a b f), (a d f), (b d f), (a b d f)
d	(a d), (b d), (a b d)
b	(a b)

<표 5> 각 클러스터로부터 얻을 수 있는 규칙들

item	첫 번째 클러스터 빈발 패턴 생성
c	(a c), (b c), (f c), (a b c), (a f c), (b f c), (a b f c)
f	(a f), (b f), (d f), (a b f), (a d f), (b d f), (a b d f)
d	(a d), (b d), (a b d)
b	(a b)

item	두 번째 클러스터 빈발패턴 생성
j	(g j), (h j), (g h j)
h	(g h)

3.2 GML에 비트 클러스터링 사용

GML 데이터의 내용은 수치값이 아닌 범주형 데이터도 있다. [14]에서도 언급하였듯이 GML 데이터는 태그 및 엘리먼트 뿐만아니라 태그 사이에 있는 데이터 값도 매우 중요하다.

<표 6> GML 내용을 테이블화 한 모습

TID	환경 정보(업종)
ST1	분식점, 호프집, 빵집
ST2	분식점, 커피숍, 패스트푸드
ST3	분식점, 호프집, 커피숍, 빵집, 패스트푸드, 서점
ST4	분식점, 패스트푸드, 커피숍, 빵집
ST5	서점, 옷가게, 신발가게, 쥬얼리, 패스트푸드
ST6	커피숍, 패스트푸드, 서점, 옷가게, 쥬얼리

<표 6>의 테이블은 각 지하철 역 주변 환경 정보를 GML데이터로 만들어놓은 것을 다시 테이블화한 모습이다. 여기서는 수치데이터가 아니라 범주형 데이터이므로 거리기반으로 유사성을 찾기에는 무리가 있다.

이것을 비트 클러스터링으로 클러스터링을 한다면 먼저 비트형태로 변환되어야 한다. 트랜잭션의 개수는 6개이고 각 업종을 아이템으로 간주하였을 때 아이템의 개수는 9개가 된다.(분식점, 호프집, 빵집, 커피숍, 패스트푸드, 서점, 옷가게, 신발가게, 쥬얼리)

<표 7> 비트 변환한 모습

TID	분식점	호프집	빵 집	커피숍	패스트푸드	서 점	옷가게	신발가게	쥬얼리
ST1	1	1	1	0	0	0	0	0	0
ST2	1	0	0	1	1	0	0	0	0
ST3	1	1	1	1	1	1	0	0	0
ST4	1	0	1	1	1	0	0	0	0
ST5	0	0	0	0	1	1	1	1	1
ST6	0	0	0	1	1	1	1	0	1

<표 8> 첫 번째 클러스터의 대푯값

TID	분식점	호프집	빵 집	커피숍	패스트푸드	서 점	옷가게	신발가게	쥬얼리
ST1	1	1	1	0	0	0	0	0	0
ST2	1	0	0	1	1	0	0	0	0
C1	1	1	1	1	1	0	0	0	0

<표 7>의 데이터를 이용하여 클러스터링을 한다면 3.1절에서 언급하였던 방법대로 각 트랜잭션들 간의 유사도와 소유 가능도 식을 이용하여 각 클러스터들의 대푯값을 구하면서 클러스터링을 하면 된다. 여기서 유사도 임계값과 소유가능도 임계값은 각각 0.5, 0.6이라고 가정한다.

<표 8>은 첫 번째 클러스터의 대푯값을 보이고 있다.

앞서 언급하였던 클러스터링 처리과정을 거치게 되면 첫 번째 클러스터(ST1, ST2, ST3, ST4)와 두 번째 클러스터(ST5, ST6)가 나오게 된다. 이와 같이 비트 클러스터링을 이용하면 범주형 비공간 데이터(속성 데이터)를 다룰 수 있게 된다.

4. 비트 클러스터링을 이용한 실험

4.1 실험 환경

본 연구의 실험 환경으로 CPU는 Pentium 4 2.4GHz, RAM 512M, 운영체제는 Windows 2000 Server, DBMS(DataBase Management System)는 MS SQL Server 2000을 사용하였으며, 프로그램 구현은 Microsoft Visual Studio .Net C#으로 하였다. 실험 데이터는 마이닝에서 자주 사용하는 chess 데이터를 이용하여 실험하였다[15]. <표 9>에 실험 환경을 정리해 놓았다.

<표 9> 실험 환경 테이블

실험 요소	실험 환경
CPU	Pentium 4 2.4GHz
OS	Windows 2000 Server
DBMS	MS SQL Server 2000
Programming Language	Microsoft Visual Studio .Net C#
실험 Data	chess Data

4.2 DB 테이블 구조

본 연구에서 실험할 때 이용한 테이블구조 중 시스템에 관련된 정보를 담는 테이블은 <표 10>과 같다.

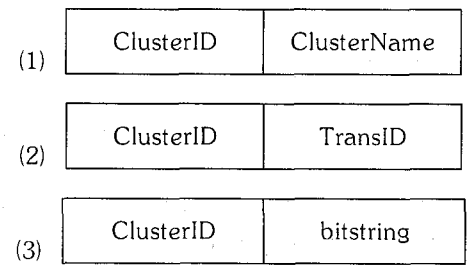
<표 10> 시스템 정보 테이블

maxcluster	itemtotal	transtotal	sim_threshold	pos_threshold	min_sup

maxcluster는 클러스터링을 할 때 클러스터 개수를 저장하기 위한 행이고, itemtotal과 transtotal은 각각 데이터베이스 내 아이템의 총 개수와

트랜잭션의 총 개수를 저장하는 행이다. sim_threshold와 pos_threshold는 비트 클러스터링 수행시 요구되는 유사도 임계값과 소유 가능도 임계값이 저장된 행이고, min_sup은 Apriori 기법 수행이나 FP-Tree 생성시 사용되는 최소 지지도 값이 저장된 행이다.

클러스터링과 관련된 테이블은 clusterTable, clusterInfoTable, clusterCenterTable이 있는데 각 테이블의 구조는 그림 1과 같다.



<그림 1> 클러스터 관련 테이블들 구조

(1)은 clusterTable로서 클러스터에 대한 정보를 담고 있는 테이블이다. (2)는 clusterInfoTable로서 각 클러스터에 어떤 트랜잭션이 있는지에 대한 정보를 담고 있는 테이블이다. (3)은 clusterCenterTable로서 각 클러스터들의 대푯값을 저장해 놓은 테이블이다.

4.3 클러스터링 비교 분석

비트 클러스터링 방법은 범주형 데이터를 다루는데 적합한 클러스터링이라고 하였다. 먼저 기존의 클러스터링 방법에서 사용하는 유사도를 구하는 방법으로 거리 기반 방법을 많이 사용하였다. 주로 많이 사용하는 것이 유클리디안 거리법이다. 또는 맨하탄이나 민코스키 거리 방법도 있다.

클러스터링 기법에는 분할 기법, 계층적 기법, 밀도 기반 기법 등이 있다. 분할 기법은 n개의 객체 또는 트랜잭션이 주어졌을 때 $k \leq n$이 되도록 k 개의 데이터로 분할을 하는 기법이다. 계층적 기법은 상향식 (bottom-up)접근 방법이거나 하향식(top-down) 접근 방법을 이용한다. 상향식 접근 방법의 경우 모든 객체 또는 트랜잭션들이 각각의 그룹을 형성해 나가게 되는데 어떠한 종료 조건까지 반복해서 진행하게 된다. 반대로 하향식 접근 방법의 경우에는 전체를 하나의 클러스터로 시작하여 여러 개의 그룹으로 어떠한 종료 조건까지 반복해서 분할해 나가는 방법이다.

밀도 기반 기법은 어떤 임의의 형태의 군집을 찾는데 밀도라는 개념에 기초하여 찾는 기법이다. 주로 공간 데이터를 다룰 때 사용한다. 여기서는 비트 클러스터링 방법과 몇 가지 대표적인 기법들과 비교해보려 한다. 대표적인 기법들 중 가장 잘 알려진 k-means 기법과 k-medoids, CLARANS 들이 있는데 이러한 클러스터링 기법들과 비교해 놓은 내용이 <표 11>에 나타나 있다.

<표 11> 각 클러스터링 기법 비교

	대푯값	비트사용	표본사용	k값 지정	거리기반
Bit-Clustering	O	O	X	X	X
k-means	X	X	X	O	O
k-medoids	O	X	X	O	O
CLARANS	O	X	O	O	O

k-means 클러스터링 기법을 제외하고 나머지 기법들은 대푯값을 사용하게 된다. k-means 클러스터링 기법은 k개의 임의의 객체들을 선택하여 모든 객체들과 평균값을 구하여 가장 가까운 평균값을 갖는 객체와 같은 클러스터에 포함되도록 한다.

비트 사용 유무는 본 논문에서 제안하는 방법인 비트 클러스터링 기법만 사용하고 있으며 표본 사용의 경우 CLARANS 알고리즘만 표본을 사용하게 되는데 이는 대용량 데이터베이스 집합에서 모든 객체 또는 트랜잭션을 다루지 않고 샘플링된 임의의 객체 또는 트랜잭션을 다루게 된다.

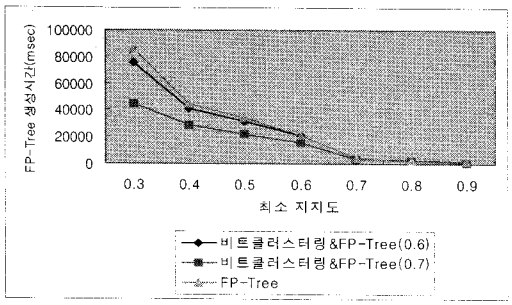
비트 클러스터링 방법을 제외하고 나머지 기법들은 모두 클러스터의 개수를 미리 정해주어야 한다. k라고 하는 것이 클러스터 개수를 k개 만들겠다라는 의미이다. 하지만 비트 클러스터링 방법은 전체 트랜잭션을 가지고 분할을 하되 정해지지 않은 개

수의 클러스터가 생성이 된다. 비트 클러스터링 방법을 제외하고 나머지 기법들은 거리 기반의 유사도 측정을 수행하여 클러스터링을 한다.

4.4 실험 내용 및 결과

본 실험에서는 마이닝에 자주 사용되는 chess 데이터를 이용하였다. 이것을 비트형태로 변환시켜 사용하였다. 실험 내용은 chess 데이터를 이용하여 FP-Tree 생성을 하였을 때와 chess 데이터를 비트 클러스터링을 이용한 후 FP-Tree를 생성하였을 때 어떤 효율성을 갖느냐이다. FP-Growth의 성능은 대부분 FP-Tree를 생성하는 것에 좌우되기 때문에 규칙을 생성하는 부분까지는 하지 않았다.

chess 데이터는 3196개의 트랜잭션과 75개의 아이템을 가지고 있다. 본 연구에서 비트 클러스터링을 수행할 때 유사도 임계값은 0.6과 0.7로, 소유 가능도 임계값은 0.6으로 고정하여 실험하였다. chess 데이터의 경우 유사도 임계값이 0.6일 때 3개, 0.7일 때 10개의 클러스터가 생성되었다.



<그림 2> 비트클러스터링 & FP-Tree 생성 실험 결과 그래프

그림 2를 통해 보면 FP-Tree를 생성하는 기존 방법보다 비트 클러스터링을 적용한 방법이 더 나은 성능을 보임을 알 수 있다. 유사도가 높은 경우 클러스터의 개수가 많아지고 클러스터에 포함되는 트랜잭션의 개수가 상대적으로 많아지게 되므로 그만큼 전체 데이터베이스를 통해 얻어지는 성능보다 나은 것을 볼 수 있다.

최소 지지도가 높아질수록 비트 클러스터링을 이용한 방법과 기존의 방법의 성능이 비슷해짐을 볼 수 있다. 최소 지지도가 높은 경우 FP-Tree 생성시 아이템 리스트에 나타나는 아이템의 개수가 그만큼 줄어들게 되고 아이템의 개수가 줄어들게 되면 당연히 가장 길이가 긴 가지는 아이템의 개수만큼 된다. 따라서 트리의 형태가 작아지게 되기 때문에 성능 차이가 크지 않다는 것이다. 따라서 적절한 최소 지지도 값을 찾는다면 성능뿐만 아니라 원하는 정보도 얻을 수 있을 것이다.

5. 결 론

본 논문에서는 비트 클러스터링을 어떻게 응용할 수 있는지 제시하였으며, FP-Growth의 문제점을 보완하기 위하여 비트 클러스터링을 사용할 것을 제안하였다. 비트 클러스터링에서는 유사도 식과 소유가능도 식을 제시하고 4가지 정의를 통해서 클러스터링을 하였다. 데이터 또는 아이템의 소유 유무에 따라 0 또는 1의 비트 값으로 바꾸어 사용하였다. 이러한 방법은 범주형의 속성 데이터나 GML 데이터 등 거리 척도로 유사성을 계산하기 힘든 데이터에 적합함을 보였다.

GML 데이터에서 사용하는 예와 FP-Growth에서 사용하는 예를 보였다. 특히 FP-Growth의 경우 FP-Tree를 만드는데 걸리는 부하와 시간에 따른 성능저하 문제를 해결하기 위하여 비트 클러스터링을 이용하는 것을 제안하였다. 실험을 통하여 비트 클러스터링을 적용한 후 FP-Tree를 생성하여 사용하는 경우에 보다 나은 성능을 보이는 것을 알 수 있었다. 다만, 최소 지지도가 높은 경우는 비트 클러스터링을 사용한 방법과 기존의 방법의 차이가 크지 않았다. 이는 최소 지지도가 높은 경우 아이템 리스트에 나타나는 아이템의 개수가 그만큼 줄어들게 되고 아이템의 개수가 줄어들게 되면 당연히 트리의 형태도 작아지게 되기 때문에 차이가 크게 나지 않는 것이다. 이렇듯 FP-Tree 생성하는데 있어서 성능을 개선하기 위하여 비트 클러스터링을 사용할 수 있다는 것을 실험을 통해 확인 할 수 있었다.

정리하자면, 비공간 데이터인 속성데이터 중에서 수치형 데이터가 아닌 범주형 데이터의 경우 비트 클러스터링을 이용하여 클러스터링 할 수 있다. GML 데이터나 다른 마이닝 방법인 연관규칙에도 응용할 수 있다. 본 논문에서는 비트 클러스터링을 GML 데이터에 어떻게 사용할 수 있는지 보였고, 연관규칙 방법 중 하나인 FP-Growth에 적용하였을 때 어떤 효과를 얻을 수 있는지 보였다. 따라서 본 연구에서는 FP-Growth에 비트 클러스터링을 적용하여 사용할 것을 제안하였다. 이는 기존의 방법보다 더 나은 성능을 갖는다는 것을 실험을 통해 확인할 수 있었으며, 이를 통하여 본 논문에서 제안하는 방법이 기존의 방법보다 우수하다는 것을 알 수 있다. 향후 연구과제로 실제 규칙을 생성하였을 때 기존 방법과의 데이터 질적 차이는 어떻게 되는지 어느 정도 차이가 나는지 실험을 통해 확인해 봐야 할 것이며, 기존의 다른 클러스터링 방법과의 비교 실험도 필요할 것이다.

참고문헌

1. M.S. Chen, J. Han, and P.S. Yu, "Data Mining: An Overview from a Database Perspective," IEEE Transactions on Knowledge and Data Engineering, Vol. 8, No. 6, December 1996, pp. 866-883.
2. A.K. Jain, M.N. Murty, and P.J. Flynn, "Data Clustering: A Review," ACM Computing Surveys, Vol. 31, No. 3, 1999, pp. 264-323.
3. K. Wang, C. Xu, and B. Liu, "Clustering Transactions Using Large Items," ACM CIKM International Conf. on Information and Knowledge Management, November 1999, pp. 483-490.
4. H. Wang, W. Wang, J. Yang, and P.S. Yu, "Clustering by Pattern Similarity in Large Data Sets," Proc. of ACM SIGMOD, Wisconsin, June 2002, pp. 394-405.
5. R. Agrawal, T. Imielinski, and A. Swami, "Mining Association Rules between Sets of Items in Large Database," Proc. of ACM SIGMOD, Washington DC, May 1993, pp. 207-216.
6. J. Han, J. Pei, and Y. Yin, "Mining Frequent Patterns without Candidate Generation," Proc. of the 2000 ACM SIGMOD International Conf. Management of Data, May 2000, pp. 1-12.
7. 김의찬, 황병연, "데이터마이닝에서 비트 트랜잭션 클러스터링을 이용한 빈발항목 생성," 한국정보처리학회논문지D, Vol. 13, No. 2, 2006
8. E.H. Han, G. Karypis, V. Kumar, and B. Mobasher, "Clustering Based On Association Rule Hypergraphs," Workshop on Research Issues on Data Mining and Knowledge Discovery, Tucson, Arizona, 1997
9. W.A. Kosters, E. Marchiori, and A.J. Oerlemans, "Mining Clusters with Association Rules," Proc. of Intelligent Data Analysis, pp. 39-50, Amsterdam, Netherlands, 1999
10. S. Guha, R. Rastogi, and K. Shim, "ROCK: a Robust Clustering Algorithm for Categorical Attributes," Proc. of the 15th International Conf. on Data Engineering, 1999
11. 김의찬, 이재민, 황병연, "XML 문서 클러스터링을 이용한 개선된 연관규칙," 제 31회 한국정보과학회 추계학술대회논문집, 제31권 제2호, 2004, pp. 181-183.
12. 김의찬, 황병연, "트랜잭션 클러스터링을 이용한 연관규칙 생성," 제 23회 한국정보처리학회 춘계학술대회논문집, 제12권 제1호, 2005, pp.15-18.
13. 김의찬, 황병연, "대용량 데이터베이스에서 클

러스터링을 이용한 빈발 패턴 생성," 제 32회 한국정보과학회 추계학술발표회논문집, 제32권 제 2호, 2005, pp. 100-102.
14. 김의찬, 황병연, "GML 문서에서 연관규칙 생성 시스템 구현," 한국공간정보시스템학회 논문지, Vol. 8, No. 1, 2006, pp. 27-35.
15. Frequent Itemset Mining Dataset Repository, http://fimi.cs.helsinki.fi/data/

김의찬
1999년 가톨릭대학교 전산학과(이학사)
2001년 가톨릭대학교 대학원 컴퓨터공학과(공학석사)
2006년 가톨릭대학교 대학원 컴퓨터공학과(공학박사)
2007년 현재 인하대학교 지리정보공학과 박사후연구원
관심분야 : 데이터마이닝, 지리정보시스템, 공간 데이터베이스, 전자상거래

김계현
1982년 한양대학교 자원공학과(공학사)
1989년 아리조나주립대 수문학과(공학석사)
1993년 위스콘신주립대 토목환경공학과(공학박사)
1995년~현재 인하대학교 지리정보공학과 교수
관심분야 : 환경, 수자원, 상하수도 및 지하시설물 관리분야의 GIS활용, 주제도 제작 및 GIS 표준화 등

이철용
2005년 인하대학교 지구환경공학부(공학사)
2007년 인하대학교 대학원 지리정보공학과(공학석사)
2007년 현재 인하대학교 대학원 지리정보공학과 박사과정
관심분야 : 데이터마이닝, 지리정보시스템, 공간 데이터베이스

박은지
2007년 인하대학교 지리정보공학과(공학사)
2007년 인하대학교 대학원 지리정보공학과 석사과정
관심분야 : 데이터마이닝, 지리정보시스템, 전자상거래